

# CMPT 280

## Topic 17: Graph Traversals

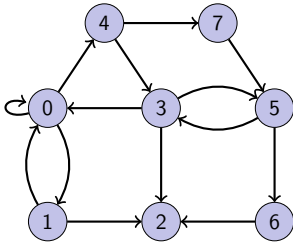
Mark G. Eramian

University of Saskatchewan

# References

- Textbook, Chapter 17

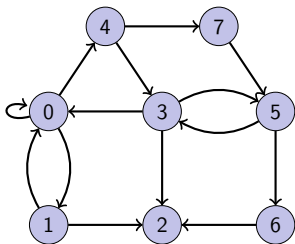
# Exercise 1



```
1 // An algorithm for breadth first
2 // traversal of a graph
3 Algorithm bft(s)
4 s is the start node in the graph
5
6 q = new Queue()
7
8 For each vertex v of V
9     reached(v) = false
10
11 reached(s) = true
12 q.insert(s)
13
14 while not q.isEmpty() do
15     w = q.item() // get top node on stack
16     q.deleteItem() // pop the stack
17
18     // perform the "visit" operation on w
19
20     For each v adjacent to w do
21         if not reached(v)
22             reached(v) = true
23             q.insert(v)
```

Find the breadth-first traversal of the graph if we assume an adjacency list representation where nodes happen to be in numerical order.

## Exercise 2



```
1 // An algorithm for depth-first
2 // traversal of a graph.
3 Algorithm dft(s)
4 s is the start node.
5
6 // V is the set of nodes in the graph
7 For each vertex v in V
8     reached(v) = false
9
10 dftHelper(s);
11
12 // Recursive helper method for algorithm dft()
13 Algorithm dftHelper(v)
14 v is a graph node
15
16 reached(v) = true
17
18 // perform the visit operation for v
19
20 For each node u adjacent to v
21     if not reached(u)
22         dftHelper(u)
```

Find the depth-first traversal of the graph if we assume an adjacency list representation where nodes happen to be in numerical order.

## Exercise 3

ajaList:  $O(|E| + |V|)$  worse =  $O(|V|^2 + |V|)$

ajaMatrix:  $O(|V|^2)$

```
1 // An algorithm for breadth first
2 // traversal of a graph
3 Algorithm bft(s)
4 s is the start node in the graph
5
6 q = new Queue()
7
8 For each vertex v of V
9     reached(v) = false
10
11 reached(s) = true
12 q.insert(s)
13
14 while not q.isEmpty() do
15     w = q.item()    // get top node on stack
16     q.deleteItem()  // pop the stack
17
18     // perform the "visit" operation on w
19
20     For each v adjacent to w do
21         if not reached(v)
22             reached(v) = true
23             q.insert(v)
```

Assuming adjacency list representation, what is the worst-case time complexity of BFT? What if we assume adjacency matrix instead?

## Exercise 4

**ajalist:  $O(\text{outd}(0)) + O(\text{outd}(1)) \dots O(\text{outd}(v))$**

```
1 // An algorithm for depth-first
2 // traversal of a graph.
3 Algorithm dft(s)
4 s is the start node.
5
6 // V is the set of nodes in the graph
7 For each vertex v in V
8     reached(v) = false
9
10 dftHelper(s);
11
12 // Recursive helper method for algorithm dft()
13 Algorithm dftHelper(v)
14 v is a graph node
15
16 reached(v) = true
17
18 // perform the visit operation for v
19
20 For each node u adjacent to v
21     if not reached(u)
22         dftHelper(u)
```

Assuming adjacency list representation, what is the worst-case time complexity of DFT? What if we assume adjacency matrix instead?

## Exercise 5

- In many applications, we associate additional data with a node besides its index.
- A specific type of graph traversal is a search for a particular node with a particular property, i.e. *breadth-first search* and *depth-first search*.
- A breadth-first or depth-first search is just a breadth-first or depth-first traversal, respectively, where we stop as soon as we find the node we are looking for and return it.
- How can we modify the breadth- and depth-first traversals to do this?

Exercise 6 `path = dfHelper(s)`  
`return path`

`path = null`

`foundV = v`

- How can we further modify the depth-first search to return, instead of the sought node, the **entire path** from the start node to the sought node?

`path = v`  
`return path`

- What is the time complexity of our resulting algorithm?

`if(!isReached(u)`

`foundV = u`

`if(path != null)`

`path = u+path`

`return path`



## Next Class

- **Next class reading:** Chapter 18: Shortest Path Algorithms for Graphs